	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorio de Computacion Salas A y B

Profesor(a): Rene Adrian Davila Perez

Asignatura: Programacion orientada a objetos

Grupo: 01

No de practica(s): 00

Integrante(s): 322001549, 322006829, 322125533, 425131152, 322103351

No de lista o brigada: 07

Semestre: 2026-1

Fecha de entrega: Lunes 18 de agosto de 2025

Observaciones:

Calificacion:

Índice

1. Introducción

1.1. Planteamiento del Problema

La documentación es un pilar fundamental en la vida profesional de cualquier ingeniero, tanto en entornos académicos como laborales. Sin embargo, es común enfrentar desafíos al dar formato a reportes técnicos o al gestionar archivos en proyectos colaborativos. Para abordar estas dificultades, herramientas como LaTeXy **Git** se han convertido en aliados indispensables.

1.2. Motivación

Por un lado, LaTeXpermite crear documentos estructurados, elegantes y profesionalmente presentados, ideal para investigaciones y artículos técnicos. Por otro, **Git** ofrece un robusto control de versiones, facilitando el trabajo en equipo al rastrear cada cambio en el código y garantizando una gestión ordenada del proyecto. Es por estas razones que se debe tener un entendimiento por lo menos básico de estas dos herramientas.

1.3. Objetivos

- Tener entendimiento acerca del uso de LaTeX.
- Comprender la importancia de **Git** al usarlo en trabajos colaborativos.
- Fortalecer las habilidades en documentación técnica y gestión de proyectos, a través del uso de **Git** y LaTeX.

2. Marco teórico

2.1. Github

GitHub:Plataforma basada en la nube donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código.

Almacenar tu código en un repositorio en GitHub te permite lo siguiente:

Presentar o compartir el trabajo. Seguir y administrar los cambios en el código a lo largo del tiempo. Dejar que otros usuarios revisen el código y realicen sugerencias para mejorarlo. Colaborar en un proyecto compartido, sin preocuparse de que los cambios afectarán al trabajo de los colaboradores antes de que esté listo para integrarlos. El trabajo colaborativo, una de las características fundamentales de GitHub, es posible gracias al software de código abierto Git, en el que se basa GitHub.[1]

2.2. Git

Git es un sistema de control de versiones que realiza un seguimiento de los cambios en los archivos. Git es especialmente útil cuando un grupo de personas y tú estáis haciendo cambios en los mismos archivos al mismo tiempo.[1]

2.3. LaTex

LaTeX es un sistema de composición tipográfica de alto nivel, diseñado para la creación de documentos científicos y técnicos con precisión profesional. A diferencia de los procesadores de texto convencionales (ej. Microsoft Word), que operan bajo un enfoque WYSIWYG ("What You See Is

What You Get"), LaTeX adopta un paradigma basado en marcado lógico, donde el usuario define la estructura semántica del contenido (secciones, ecuaciones, referencias) y el motor de LaTeX se encarga automáticamente del formato, garantizando resultados estéticamente impecables.[2]

2.4. Public class main y Public static void main(String[] args)

La palabra clave `public` adjunta a la declaración de clase no es una elección trivial, sino una decisión de diseño deliberada que garantiza la accesibilidad. Al marcar la clase como pública, los desarrolladores de Java indican a la JVM y a otras partes del programa que esta clase es accesible desde cualquier otra clase, independientemente del paquete al que pertenezca. Esto es fundamental porque la JVM necesita acceso sin restricciones a esta clase para iniciar la aplicación.[3]

Dentro de `public class Main`, reside un método que actúa como punto de entrada para la aplicación: `public static void main(String[] args)`. Este método es tan vital que la JVM lo busca específicamente al iniciar una aplicación Java. Analicemos sus componentes para comprender sus funciones individuales y colectivas:

- `public`: Al igual que con la clase, esta palabra clave hace que el método sea accesible desde cualquier otra parte del programa o incluso desde otros programas. Este acceso universal es necesario porque el método principal debe poder ser invocado por la JVM, que se encuentra fuera del ámbito del paquete de la aplicación.[3]
- `static`: Esta palabra clave permite llamar al método sin tener que instanciar la clase. Se trata de un requisito práctico, ya que el método principal debe ejecutarse para iniciar el programa, y crear un objeto de la clase antes de ejecutar cualquier método daría lugar a una situación paradójica en la que se necesitaría que el programa se ejecutara para iniciarse a sí mismo.[3]
- `void`: En Java, los métodos pueden devolver valores, pero el método principal no. Esto se indica con la palabra clave void, que significa que, tras completar su ejecución, el método principal no devuelve ningún valor.[3]
- `main`: El nombre del método en sí no es arbitrario, sino una convención que espera la JVM. Busca específicamente un método con este nombre exacto para iniciar la ejecución del programa.[3]
- `String[] args`:Este parámetro representa una matriz de objetos String, comúnmente conocidos como argumentos de línea de comandos. Se trata de argumentos que se pueden pasar al programa en el momento de la ejecución, lo que permite influir en el comportamiento del programa pasando diferentes valores.[3]

2.5. Funcionamiento

La ejecución de un programa Java es un proceso meticulosamente orquestado que implica varias etapas, cada una de ellas crucial para el funcionamiento fluido de su aplicación. Comprender estas etapas no solo aclara lo que sucede cuando se ejecuta un programa Java, sino que también mejora su capacidad para escribir código Java más eficiente y eficaz.[3]

1. *Cargando*: El proceso comienza cuando la JVM carga la clase `Main`. Este paso no es tan sencillo como podría parecer, ya que implica buscar el archivo de clase, leer sus datos binarios y crear un objeto `Class` en el área de métodos de la JVM. Este objeto contiene metadatos sobre la clase, como su nombre, métodos y variables. El subsistema del cargador de clases desempeña aquí un papel fundamental, ya que garantiza que la clase solo se cargue una vez para mantener la alta seguridad y la eficiencia en tiempo de ejecución de Java.[3]

2. *Vinculación*: Una vez cargada, la clase se somete a un proceso de vinculación, que implica verificación, preparación y resolución opcional:
    - Verificación: Garantiza la corrección del archivo de clase, comprobando si hay errores estructurales, sintácticos y semánticos para evitar errores de ejecución y violaciones.[3]
    - Preparación: Asigna memoria para las variables de clase y las inicializa con valores predeterminados, preparando el escenario para la inicialización explícita.[3]
    - Resolución (opcional en esta etapa): Transforma las referencias simbólicas del archivo de clase en referencias directas utilizadas por la JVM, un paso que también se puede aplazar hasta su uso real (resolución diferida o tardía).[3]
3. *Inicialización*: La inicialización da vida a la clase, pasando de los valores predeterminados a las inicializaciones explícitas. Se establecen las variables estáticas y se ejecutan los bloques estáticos en el orden en que aparecen en la clase. Esta etapa es crucial para configurar el entorno en el que se ejecutará el método `main`, asegurando que se cumplan todas las condiciones y requisitos previos necesarios.[3]
4. *Ejecución*: Una vez preparado el escenario, la JVM se centra en el método `main`, el punto de entrada de su aplicación Java. El método `main` es único porque es el puente entre el inicio de la ejecución y la lógica de la aplicación. Opera dentro del entorno inicializado, accediendo a variables estáticas e invocando otros métodos para realizar las tareas de la aplicación. La forma en que está estructurado el método `main` influye en el flujo de toda la aplicación, lo que lo convierte en un componente crítico de la programación Java.[3]
5. *Limpieza*:Una vez completado el método `main`, el programa no se detiene, sino que se inicia una fase de limpieza. La JVM inicia la recolección de basura, recuperando la memoria asignada a los objetos que ya no se utilizan. Esta fase es crucial para la gestión de la memoria, ya que garantiza que los recursos se utilicen de manera eficiente y se liberen cuando ya no sean necesarios. En la mayoría de las aplicaciones, esta fase es gestionada automáticamente por la JVM, lo que evita al desarrollador la complejidad de la gestión de la memoria.[3]

3. Desarrollo

3.1. Descripción de la Implementación

Para esta práctica se trabajó con el lenguaje de programación Java. Como primer paso fue necesaria la instalación del entorno que permite compilar y ejecutar programas en este lenguaje. Una vez configurado, se procedió a crear un archivo con extensión .java denominado HolaMundo.java.

El programa implementado fue un ejemplo introductorio donde se aplicaron los conceptos básicos vistos en clase, como:

- La definición de una clase principal.
- La inclusión de un método principal (main), que es el punto de inicio de cualquier aplicación en Java.
- La instrucción encargada de mostrar un mensaje en pantalla.

De esta manera se aseguró que el entorno de desarrollo estuviera correctamente instalado y que el compilador reconociera la estructura mínima de un programa en Java.

Finalmente, el archivo fue organizado dentro de una carpeta llamada Reto y subido a un repositorio de GitHub, con lo cual se reforzó el uso de control de versiones y el manejo de repositorios remotos.

3.2. Pruebas Realizadas

Las pruebas consistieron en ejecutar el archivo de HolaMundo.java creado para darnos una introducción a java como se suele realizar con cada lenguaje de programación.

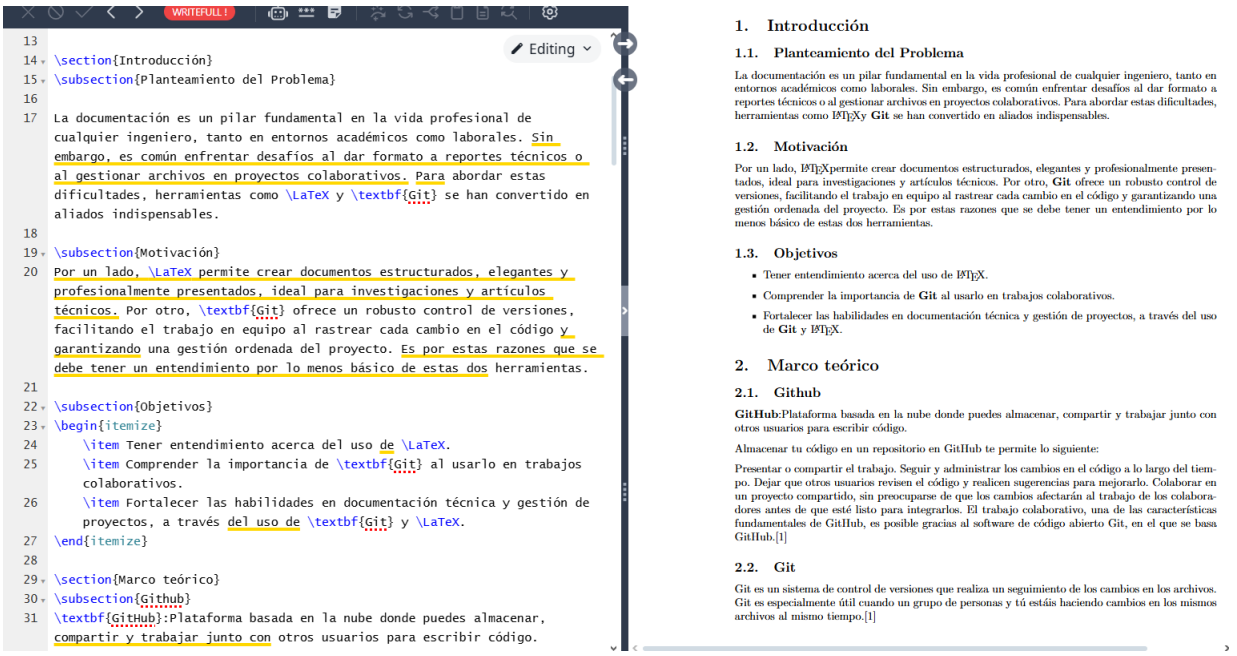
- **class:** palabra reservada para definir una clase en Java.
- **public:** modificador de acceso → significa que la clase puede ser usada desde cualquier otro lugar del programa.
- **static:** significa que el método pertenece a la clase y no a un objeto creado a partir de ella (así el sistema puede ejecutarlo sin necesidad de instanciar la clase).
- **void:** tipo de retorno → aquí significa que no devuelve ningún valor.
- **main:** nombre especial del método que Java busca para empezar a ejecutar el programa.
- **String[] args:** es un arreglo de cadenas que puede recibir argumentos desde la línea de comandos.
- **System:** clase integrada de Java que ofrece acceso al sistema (entrada, salida, errores, etc.).
- **out:** es un objeto dentro de System que representa la salida estándar (generalmente la consola).
- **println:** es un método que imprime el texto en pantalla y hace un salto de línea al final.

Cuando se ejecuta el codigo muestra en la pantalla el "Hola Mundo desde Java".

Posteriormente se subieron los archivos a GitHub desde la terminal. Para conectar el repositorio con GitHub lo que se realizo fue usar el comando **"git remote add origin" + URL**.

Para guardar y subir los cambios se hizo uso del comando **"git add ."** que añade todos los archivos al stanging (area de preparacion), luego se utilizo el comando **git commit -m "mensaje de los cambios o de lo que se hizo"** y por ultimo se utilizaron los comandos **"git branch -M main"** y **"git push -u origin main"**, git branch es para generar una copia del proyecto que puedes modificar sin afectar la rama principal (proyecto original), por otro lado, git push ya realiza los cambios que afectan la rama principal.

4. Resultados



```
\section{Introducción}
\subsection{Planteamiento del Problema}

La documentación es un pilar fundamental en la vida profesional de
cualquier ingeniero, tanto en entornos académicos como laborales. Sin
embargo, es común enfrentar desafíos al dar formato a reportes técnicos o
al gestionar archivos en proyectos colaborativos. Para abordar estas
dificultades, herramientas como \LaTeX y \textbf{Git} se han convertido en
aliados indispensables.

\subsection{Motivación}
Por un lado, \LaTeX permite crear documentos estructurados, elegantes y
profesionalmente presentados, ideal para investigaciones y artículos
técnicos. Por otro, \textbf{Git} ofrece un robusto control de versiones,
facilitando el trabajo en equipo al rastrear cada cambio en el código y
garantizando una gestión ordenada del proyecto. Es por estas razones que se
debe tener un entendimiento por lo menos básico de estas dos herramientas.

\subsection{Objetivos}
\begin{itemize}
    \item Tener entendimiento acerca del uso de \LaTeX.
    \item Comprender la importancia de \textbf{Git} al usarlo en trabajos
    colaborativos.
    \item Fortalecer las habilidades en documentación técnica y gestión de
    proyectos, a través del uso de \textbf{Git} y \LaTeX.
\end{itemize}

\section{Marco teórico}
\subsection{Github}
\textbf{GitHub}:Plataforma basada en la nube donde puedes almacenar,
compartir y trabajar junto con otros usuarios para escribir código.
```

1. Introducción

1.1. Planteamiento del Problema

La documentación es un pilar fundamental en la vida profesional de cualquier ingeniero, tanto en entornos académicos como laborales. Sin embargo, es común enfrentar desafíos al dar formato a reportes técnicos o al gestionar archivos en proyectos colaborativos. Para abordar estas dificultades, herramientas como LaTeXy **Git** se han convertido en aliados indispensables.

1.2. Motivación

Por un lado, LaTeXpermite crear documentos estructurados, elegantes y profesionalmente presentados, ideal para investigaciones y artículos técnicos. Por otro, **Git** ofrece un robusto control de versiones, facilitando el trabajo en equipo al rastrear cada cambio en el código y garantizando una gestión ordenada del proyecto. Es por estas razones que se debe tener un entendimiento por lo menos básico de estas dos herramientas.

1.3. Objetivos

- Tener entendimiento acerca del uso de LaTeX.
- Comprender la importancia de **Git** al usarlo en trabajos colaborativos.
- Fortalecer las habilidades en documentación técnica y gestión de proyectos, a través del uso de **Git** y LaTeX.

2. Marco teórico

2.1. Github

GitHub:Plataforma basada en la nube donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código.

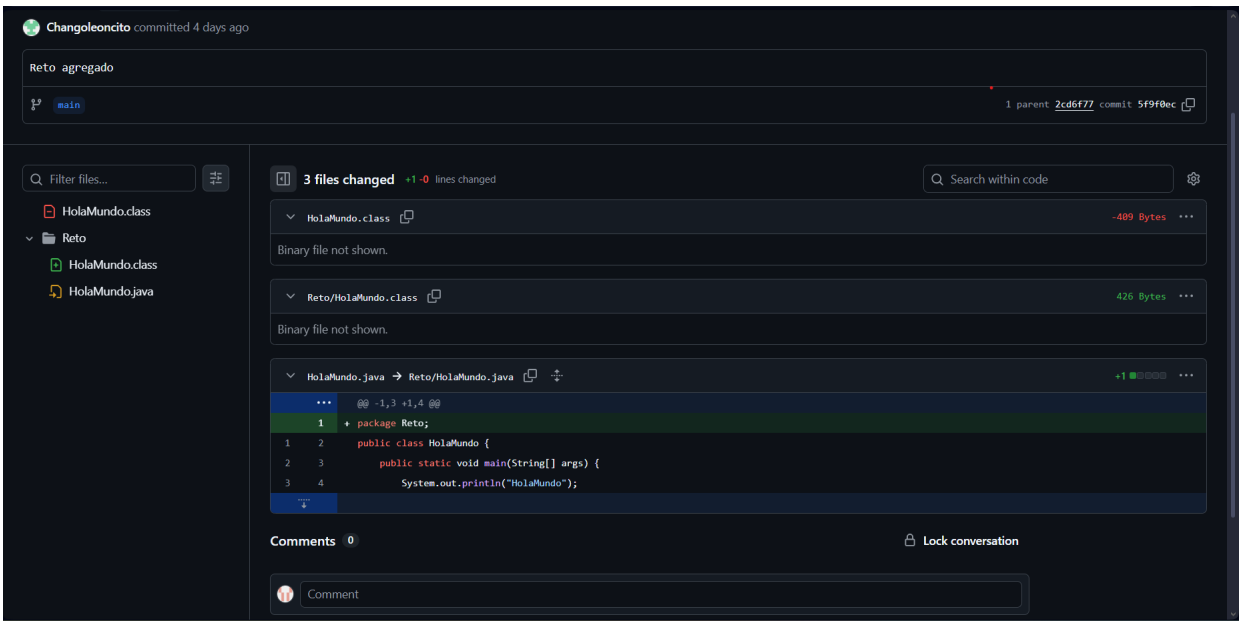
Almacenar tu código en un repositorio en GitHub te permite lo siguiente:

Presentar o compartir el trabajo. Seguir y administrar los cambios en el código a lo largo del tiempo. Dejar que otros usuarios revisen el código y realicen sugerencias para mejorarlo. Colaborar en un proyecto compartido, sin preocuparse de que los cambios afectarán al trabajo de los colaboradores antes de que esté listo para integrarlos. El trabajo colaborativo, una de las características fundamentales de GitHub, es posible gracias al software de código abierto Git, en el que se basa GitHub.[1]

2.2. Git

Git es un sistema de control de versiones que realiza un seguimiento de los cambios en los archivos. Git es especialmente útil cuando un grupo de personas y tú estáis haciendo cambios en los mismos archivos al mismo tiempo.[1]

En la imagen se muestra como se le dio uso a LaTeXpara la creación de un documento PDF, mismo que se muestra a la derecha y que es el que se esta observando. También se puede apreciar como se le dio uso a las herramientas dentro de LaTeXpara dar un mejor formato, más elegante y preciso al documento.



En la imagen se muestra la creación de el repositorio, ademas de un archivo ya subido y guardado en el **"GIT"**.

5. Conclusiones

El ejercicio de implementar un programa básico como Hola Mundo permite comprender la estructura esencial de un proyecto en este lenguaje y verificar la correcta instalación del entorno. Ademas, la vinculación entre Visual Studio Code y GitHub fortalece las buenas prácticas de programación al facilitar la organización, control de versiones y colaboración en proyectos.

Referencias

[1] GitHub. (2025) Acerca de GitHub y Git. [Accedido: 18-ago-2025]. [Online]. Available: https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

[2] L. Lamport. (1994) LaTeX: A Document Preparation System. Accedido: 15-ago-2025. [Online]. Available: https://lamport.azurewebsites.net/pubs/pubs.html

[3] A. Obregón. (2024) Beginner's guide to 'public class main' in java. Accedido: 15-ago-2025. [Online]. Available: https://medium.com/@AlexanderObregon/beginners-guide-to-public-class-main-in-java-c6dc45e1b6af